

Tópicos Abordados



- O formato XML
- API DOM
- LINQ para XML
 - Principais classes
 - Criando um documento XML
 - Lendo e gravando documentos XML
 - Manipulando dados do XML
- A classe XmlReader

O Formato XML



- E<u>x</u>tensible <u>M</u>arkup Language
- Padronizado pelo W3C
 - World Wide Web Consortium
- É um formato para representação de dados
 - Hierárquico
 - Padrão aberto e portável
- Muito usado atualmente
 - Integrações de dados entre sistemas
 - Web services
 - Arquivos de configuração
 - Representação dos mais diversos tipos de informações

A API DOM



- **D**ocument **O**bject **M**odel
- API especificada pelo W3C
- Permite manipular a estrutura do XML
- Esta API cria uma árvore em memória que representa e estrutura do documento XML
 - É preciso ficar atento ao consumo de memória
- Diversas linguagens de programação suportam o uso de DOM
 - No C#, o DOM é suportado através do namespace System.Xml
 - Atualmente, o LINQ para XML é uma alternativa mais interessante

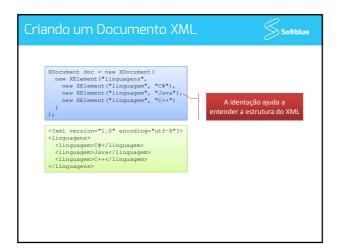
LINQ para XML



- O LINQ para XML está disponível através do namespace *System.Xml.Linq*
- É possível gerar documentos XML de forma mais simples do que usando DOM
- A extração de dados do XML pode ser feita através do uso de LINQ

Principais Classes de System.Xml.Linq Sontblue

Classe	O que representa
XDocument	Documento XML
XElement	Elemento (tag) dentro do XML
XAttribute	Atributo de um elemento
XDeclaration	Declaração no início do XML
XComment	Comentário
XCData	Seção CDATA de uma tag
XNamespace	Namespace



```
Criando um Documento XML

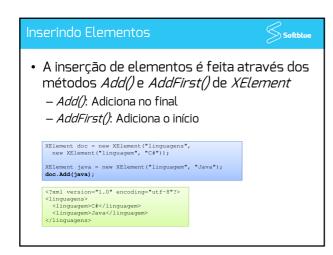
XDocument doc = new XDocument(
    new XElement("linguagem",
    new XElement("linguagem",
    new XElement("inguagem",
    new XElement("inguagem",
    new XElement("nome", "Java"),
    new XElement("autor", "Java"),
    new XElement("nome", "Ot+"),
    new XElement("nome", "C++"),
    new XElement("descricao", new XCData("Linguagem orientada a objetos"))
};

<?xml version="1.0" encoding="utf-8"?>
<!inguagem ano="2001">C$</iinguagem>
<!inguagem>
<!inguagem>
<!utorJames Gosling</autor>
<!linguagem>
<!nome>C++</nome>
</descricao><!/DATA[Linguagem orientada a objetos]]></descricao>
</linguagem>
namayougemnamayougem>
namayougem>
namayougem
```



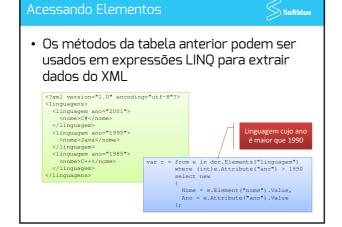


• As classes XDocument e XElement possuem os métodos Load(), Parse() e Save() doc.Save(@"C:\Temp\arquivo.xml"); XElement e = XElement.Load(@"C:\Temp\arquivo.xml"); StringWriter w = new StringWriter(); doc.Save(w): string s = ...; XElement e = XElement.Parse(s);



Pemovendo Elementos A remoção é feita através do método Remove() de XElement - Pode ser usado para remover um elemento ou um fragmento completo do XML (árvore) | XElement doc = new XElement ("linguagens", new XElement ("linguagem", "C#"), new XElement ("linguagem", "Java"), new XElement ("linguagem", "C++"); | doc.Elements ("linguagem") .Remove(); | <?xml version="1.0" encoding="utf-8"?>

• LINQ para XML possui alguns métodos para acessar elementos do XML – Definidos em *Extensions* como extension methods de IEnumerable<T> Método O que retorna Attributes() Atributos do elemento Attribute() Atributo do elemento com determinado nome Descendants() Elementos dentro de um elemento (descendentes em qualquer nível) Elements() Elementos dentro de um elemento (descendentes diretos) Element() Primeiro elemento descendente Ancestors() Pais de um elemento



A Classe *XmlReader*



- Existem situações onde a geração da árvore do XML feita pelo DOM não é necessária ou não é viável
 - Documentos XML muito extensos
- Uma alternativa é usar a classe *XmlReader*
 - Considerado um *pull parser*
 - O XML pode ser processado enquanto é lido
 - Não existe a criação da árvore em memória
 - A navegação no XML não é tão flexível quanto no DOM
- A classe faz parte do namespace System.Xml

Exemplo de Uso do XmlReader



<?xml version="1.0" encoding="utf-8"?>
<linguagems>
<linguagem ano="2001">
</nome>C*</nome>
</linguagem>
<linguagem ano="1995">
<nome>Java</nome>
</linguagem>
<linguagem ano="1995">
<nome>Java</nome>
</linguagem>
</nome>
</linguagem>
</nome>
</linguagem>
</nome>
</linguagem>
</nome>
</linguagem>
</nome>
</linguagem>
</nome>
</linguagem>
</linguagem>
</linguagem>
</linguagem>
</linguagem>
</linguagens></linguagens></linguagens></linguagens></or>

XmlReader reader = XmlReader.Create(@*C:\arquivo.xml*);
while (reader.Read())
{
 if (reader.NodeType == XmlNodeType.Element && reader.Name == "linguagem")
 {
 int ano = int.Parse(reader.GetAttribute("ano"));
 }
 else if (reader.NodeType == XmlNodeType.Element && reader.Name == "nome")
 {
 String nome = reader.ReadElementContentAsString();
 }
}

